REMARKS

Claims 2-8 are pending in the present application, claims 7-8 having been added and claim 1 having been cancelled herein. The Office Action and cited references have been considered. Favorable reconsideration is respectfully requested.

Claims1-3 were objected to because of a number of grammatical errors. The claims have been amended to overcome this objection. Withdrawal of this objection is respectfully requested.

Claims 1-6 were rejected under 35 U.S.C. §112, second paragraph. The claims have been amended to overcome this rejection. Withdrawal of this rejection is respectfully requested.

Claims 1-6 were rejected under 35 U.S.C. §103 as being unpatentable over Baentsch (U.S. Patent No. 6,792,612) ("Baentsch '612) in view of Baentsch (U.S. Patent Application No. 2002/0093856) ("Baentsch '856). These rejections are respectfully traversed for the following reasons.

Claim 7 recites a method for loading into a computer device with a programming language using objects, an updated release of an earlier application having earlier application classes and earlier static field identifiers, the updated release having updated application release classes, and updated static field identifiers, the programming language permitting an introduction of additional classes, a class hierarchy modification and a definition of further fields and methods. The method includes the steps of computing, in a first computing operation prior to the loading, a class matching information establishing a correspondence between the earlier application release classes and the

updated application release classes, computing, in a second computing operation prior to the loading, a second static field identifiers matching information establishing a correspondence between the earlier application release static field identifiers and the updated application release static field identifiers, linking the class matching information and the static field identifiers matching information to the updated application release as loaded into the computer device; and using the class matching information and the static field identifiers matching information to modify the objects to point at the updated application release classes and use the updated application release static field identifiers. This is not taught, disclosed or made obvious by the prior art of record.

The previous claim 1 has been cancelled and is now replaced by the new claim 7. Accordingly, claims 2-6 now refer to the claim 7 instead of claim 1. A new claim 8 has been added. Applicant respectfully submits that these amended claims clarify that Applicant's invention concerns a method for updating one application already loaded in a computer device (*i.e.*, modification of existing code, and not loading of new code).

More particularly, new claim 7 is directed to a method for loading an updated release of an application in which the existing code and data are modified in order to link them to the updated loaded release of the application. This claimed method is quite different from the method disclosed by Baentsch '612.

Baentsch '612 teaches a method for introducing new code into a runtime Java® system, and modifying the newly loaded code in order to allow it to run on the runtime Java® system.

The goals of Baentsch '612 and the present invention are completely different. In particular, Baentsch et al. teaches a method to load a new application, while claim 7 concerns a method to update an existing application.

The concepts of Baentsch '612 and the present invention are completely different. In particular, Baentsch '612 innovate by not linking the newly loaded application in an innovative way. Applicant proposes a modification of the pre-existing code and data in an innovative way. These two concepts have only in common the fact that they are both related to the linking of two pieces of code, which is in both cases performed using specific link tables, as an ordinarily skilled person in the art would understand.

Moreover, there are differences in the features of Applicant's claimed invention and the method disclosed by Baentsch '612. In particular, Applicant's recites a method to load an updated release of an application, while Baentsch '612 discloses a method for introducing new code into a runtime Java® system, which is the classical application loading scheme, which does not address the update of existing code.

Applicant's claim 7 states that the method permits an introduction of additional classes, a class hierarchy modification, and a definition of further fields and methods. In contrast, Baentsch '612 teaches that the programmer can freely add private and static method and classes, which is very different. Baentsch '612 does not cover the class hierarchy modification aspect of Applicant's invention, as it does not allow the modification of existing code.

Applicant's claim 7 recites the step of, *inter alia*, "linking said class matching information and said static field identifiers matching information to said updated

application release as loaded into the computer device". In contrast, Baentsch '612 discloses that "there is one fixup table,... and the link process is described," where the processes are fundamentally different in the sense that Baentsch '612 teaches a way to update the newly loaded cardlet, whereas Applicant's claimed method teaches a way to modify the pre-existing application already loaded on the device.

Applicant's method as claimed in claim 7 comprises a step of "using said information to modify the objects". With respect to this step, the Examiner asserts that "figure 3 of Baentsch et al shows the symbolic binding of an applet against a target package." This assertion is not pertinent due to the fact that, as would be understood by one of ordinary skill in the art, a target package refers to code, whereas objects refer to data, which are handled in different ways. This is a key point of Applicant's invention, in which pre-existing data is modified in order to take into account the newly loaded code.

Baentsch '856 concerns a method for language verification of a Java® Card CAP file created from an original Java® code file comprising a conversion step for converting the Java® Card CAP file into a corresponding converted Java® code file that is semantically identical to the Java® Card CAP file, and a language verification step for verifying the converted Java® code file for compliance with Java® language specifications. Applicant respectfully submits that the purpose of this method is quite different from that of Applicant's method. For this reason, the teaching of Baentsch '856 would not have been used by one of ordinary skill in the art to solve the problems faced by Applicant.

Baentsch '856 was cited only because it teaches a step of computing prior to a step of loading. Because the Office Action cites 5 full paragraphs as allegedly teaching this

step, Applicant can only assume that, particular, the Examiner refers to "the interpreted nature of the language and thus inherent continuous checks of the Java® code prior to its actual execution" since the rest of the cited paragraphs do not appear to be at all relevant. However, Applicant is not claiming "computing prior to loading" *per se* in claim 7. The fact that some computing has to be performed prior to loading is usual. However, Applicant respectfully submits that the particular information being computed according to Applicant's claims are not shown in the prior art as being computed prior to loading an updated release of an earlier application. In particular, such a check of Java® code, as mentioned in Baetsch "856, does not correspond to Applicant's steps of computing, prior to loading an updated application release, a piece of information for matching the classes and the static field identifiers of the earlier application release to the classes and the static identifiers of the updated application release as recited in claim 7.

For these reason the combination of teachings of Baentsch '612 and Baentsch '856 does not lead to Applicant's claimed solution. For at least these reasons, Applicant respectfully submits that claim 7 is patentable over the prior art of record whether taken alone or in combination as proposed in the Office Action. Claims 2-6 and 8 are believed to be patentable in and of themselves, and as they depend from and include the limitations of claim 7, which is patentable for the reasons discussed above.

In view of the above amendment and remarks, Applicant respectfully requests reconsideration and withdrawal of the outstanding rejections of record. Applicant submits that the application is in condition for allowance and early notice to this effect is most earnestly solicited.

If the Examiner has any questions, he is invited to contact the undersigned at 202-628-5197.

Respectfully submitted,

BROWDY AND NEIMARK, P.L.L.C.
Attorneys for Applicant(s)

By /Ronni S. Jillions/
Ronni S. Jillions
Registration No. 31,979

RSJ:me
Telephone No.: (202) 628-5197
Facsimile No.: (202) 737-3528
G:\bn\m\mout\Vetillard1\pto\2008-07-03AmendmentVETILLARD1.doc